Joint Classification and Trajectory Regression of Online Handwriting using a Multi-Task Learning Approach

Felix Ott[1,2] David Rügamer[2] Lucas Heublein[1] Bernd Bischl[2]
Christopher Mutschler[1]
[1]Fraunhofer IIS, Fraunhofer Institute for Integrated Circuits IIS
[2]LMU Munich, Munich, Germany
{felix.ott, christopher.mutschler}@iis.fraunhofer.de
{david.ruegamer, bernd.bischl}@stat.uni-muenchen.de

Abstract

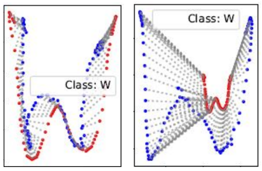
Multivariate Time Series (MTS) classification is important in various applications such as signature verification, person identification, and motion recognition. In deep learning these classification tasks are usually learned using the cross-entropy loss. A related yet different task is predicting trajectories observed as MTS. Important use cases include handwriting reconstruction, shape analysis, and human pose estimation. The goal is to align an arbitrary dimensional time series with its ground truth as accurately as possible while reducing the error in the prediction with a distance loss and the variance with a similarity loss. Although learning both losses with Multi-Task Learning (MTL) helps to improve trajectory alignment, learning often remains difficult as both tasks are contradictory. We propose a novel neural network architecture for MTL that notably improves the MTS classification and trajectory regression performance in online handwriting (OnHW) recognition. We achieve this by jointly learning the cross-entropy loss in combination with distance and similarity losses. On an OnHW task of handwritten characters with multivariate inertial and visual data inputs we are able to achieve crucial improvements (lower error with less variance) of trajectory prediction while still improving the character classification accuracy in comparison to models trained on the individual tasks.

1. Introduction

MTS Classification (MTSC) identifies labels based on MTS as a set of real-valued, sequentially ordered observations. MTSC appears across many domains, e.g., human activity recognition (HAR), handwriting reconstruction, and medical data analysis. [31, 46, 60, 62] A related yet different task is trajectory reconstruction and function alignment. This is important to applications that involve the modeling of mathematical functions or for shape analysis, e.g., to optimally transform a shape into another shape [19, 53].

An application where both tasks must be solved simultaneously is HAR [1, 4]. For example, the control of smart devices through hand-movement patterns or sport applications [35] requires a joint learning of the pattern classification and the hand trajectory. The data are recorded using a handheld device with inertial sensors or by outside-in cameras, e.g., a Kinect system. For our handwriting recognition application, common techniques require to write on a device where the writing style is influenced, to take images of the handwritten text, or to use a stylus pen, a touch pen with a sensible magnetic mesh tip together with a touch screen surface [2]. Systems for writing on paper are only prototypical, such as the ones used in [12, 50, 58], or are smartphones that provide a pen-like interaction from standard built-in sensors [17]. For our OnHW application, we used a novel sensor-enhanced pen [33, 46] and recorded the pen movement with outside-in cameras.

Combined metrics. In computer vision tasks such as landmark localization [9] and human pose estimation [41], DL has successfully contributed to sequence to sequence regression-based methods [15]. However, in trajectory prediction we do not only want to align reconstructed trajectories with their ground truth, but also require them to be smooth over time [59]. A combination of distance metrics with geometric shape-based and spatio-temporal metrics achieves such a smoothness [34]. However, both metrics contradict each other which makes them difficult to be used together for training (cf. the blue trajec-



(a) Distance loss. (b) Similarity loss.

Figure 1: Ground truth (red) and reconstructed (blue) trajectories.

Input	Task	Loss	Output
Inertial	Classification	Cross-entropy	Character class
or		Distance	**or/and**
visual	Regression	Spatio-temporal	trajectory (MTS)
MTS		Distribution	

Table 1: Summary of the challenge addressed in this paper.

tory of the handwriting reconstruction task in Fig. 1. Optimizing a *distance* loss (here: Euclidean, Fig. 1a) introduces a relative error and dilatation while a *spatio-temporal* loss (here: Pearson Correlation, Fig. 1b) provides a smooth and similarly shaped trajectory, but with (large) scaling error.

We address the problem of learning both metrics simultaneously by MTL. MTL exploits differences and commonalities across two or more single learning tasks to solve them jointly with possibly improved performance in each single task. As the usefulness of different tasks is not known a priori, combining loss functions is one of the main challenges. The goal in MTL is to find appropriate weighting strategies such that the total loss is minimized optimally [13, 20, 40]. MTL research made significant progress over the last years regarding training techniques [44, 61], but is still challenging to apply for such contradictory tasks.

In this paper we propose different MTL architectures that combine two heterogeneous but subtly correlated tasks: MTSC and trajectory regression on two different datasets from an OnHW recognition application [46]. Table 1 summarizes the challenge we address. We use MTL [40] and show that the order of weight increase is crucial for smooth trajectory regression. Our results show an improved character classification and optimal trajectory regression by combining the cross-entropy loss with distance and similarity losses, i.e., spatio-temporal metrics.

The paper is organized as follows. We discuss related work in Sec. 2. Sec. 3 explains how we combine classification and regression loss functions in an MTL setup. Sec. 4 presents our novel OnHW datasets and CNN architectures before Sec. 5 shows experimental results. Sec. 6 concludes.

2. Related Work

We discuss related work on MTSC (Sec. 2.1), loss functions for trajectory regression (Sec. 2.2), and MTL (Sec. 2.3).

2.1. Deep Learning based Multivariate Time Series Classification (MTSC)

MTSC is used in different fields to estimate class labels based on several (in-)dependent time series. While there are many shapelet- and Fourier-based methods, we focus on DL methods as they are most similar to our approach. DL-based methods often exploit LSTM and CNN layers to extract features. Examples include the multi-channel CNN for univariate processing by [62] and the attention-based LSTM by [29]. [31] introduced a squeeze-and-excitation block to generate latent features for classification (MLSTM-FCN). [60] proposed the attentional prototype TapNet that handles the issue of limited training labels combined with a random group permutation method. An overview of MTSC methods can be found in [22]. However, as such approaches focus on classification only they perform poorly on (smooth) trajectory regression.

2.2. Trajectory Regression

For the reconstruction of time series we need a metric to measure the similarity between a predicted time series and its ground truth. We here briefly review known metrics and discuss them in the context of reconstructing trajectories.

Commonly used *distance*-based metrics are the Mean Squared Error (MSE), the Mean Absolute Error (MAE) (being more robust to outliers but with potentially large gradients near the optimum), the Huber loss [30], Andrew's Sine, and Tuckey's Biweight [7], each handling "outliers" differently [8]. Although practically relevant, these methods do not consider temporal dimensions or guarantee smoothness.

Geometric shape-based similarity measures are the Fréchet distance [10], preserving the time series order of sequence data along curves, the Hausdorff distance [55], a measure for dissimilarity for comparing point sets, and the Procrustes analysis [51]. [59] proposed a trajectory simplification by using sub-trajectory similarity information by the Fréchet and Hausdorff distances. However, optimizing such metrics is often difficult as they are not differentiable.

Spatio-temporal distance metrics take into account both the spatial and the temporal dimensions of movement data, such as time-dependent trajectories. Examples include the Cosine Similarity and Pearson Correlation [47]. However, these metrics are inappropriate for shape reconstruction being invariant to scaling and translation.

Time Warping approaches such as Dynamic Time Warping (DTW) [11, 14] can compare time series of variable size and are robust to shifts or dilatation across time. Approaches typically solve a minimal-cost alignment problem solved with dynamic programming [15] using the MSE or the Mahalanobis distance [24]. For audio-to-audio alignment [24] proposed to learn Hamming and Mahalanobis metrics. [17] used dynamic programming to compare reconstructed trajectories, but only for result evaluation. However, such methods may lead to pathological results as warping the x-axis can produce unintuitive alignments.

Distribution-based methods exploit the distributional discrepancy of samples. One important example is the Kullback-Leibler (KL) divergence [3]. Optimal Transport compares probability distributions in a geometrically faithful way, but is limited because of its computational burden (e.g., the Wasserstein distance [23]). [28] reconstructed sketches using RNNs that are, however, represented as time-independent vector images and not trajectories. As the KL

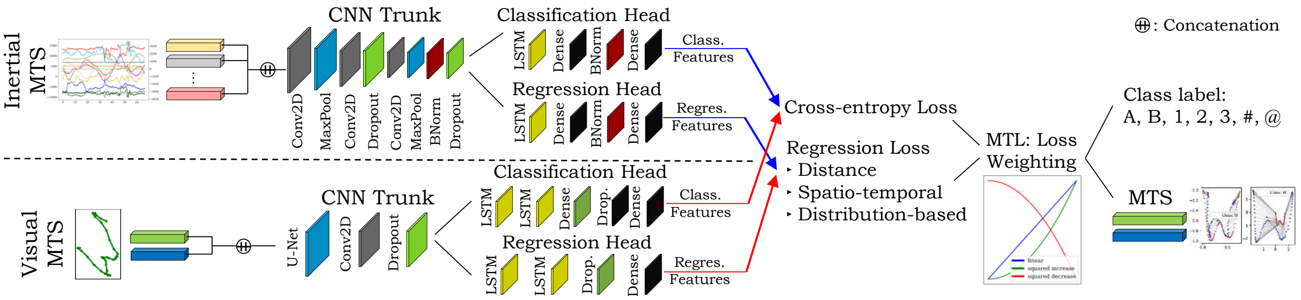


Figure 2: Method overview: MTS input, exemplary CNN trunk and heads, and class and trajectory prediction. The top row shows the architecture for the IMU (inertial measurement unit) dataset, the bottom row shows the architecture for the visual dataset. Classification (cross-entropy) and regression (distance, spatio-temporal and distribution-based) loss functions are combined with different MTL weighting strategies. $+\!\!\!+$: concatenation. The right part of the pipeline is equal for both datasets and CNNs, but the input is separate (inertial features or visual features). Different types of data are not combined.

divergence is asymmetric and does not satisfy the triangle equality, we will focus on the Wasserstein metric.

Various approaches for handwriting regression have been proposed, such as the application of adversarial domain adaptation for training generative RNNs on handwriting generation with MNIST [26], or sequence order inference by combining the $\mathcal{L}_2$ norm and the Pearson Correlation [34]. However, they are training on relative distance pairs.

2.3. Multi-Task Learning (MTL)

MTL achieves a provable information gain over single task learning if the jointly learned tasks are somehow related [16]. A naive approach combines multiple losses using a weighted sum of losses of the single tasks. However, the model performance is very sensitive to the actual weight selection. Hence, [32] considered the homoscedastic uncertainty of each task to weight multiple tasks differently. [61] addressed the problem of learning heterogeneous but subtly correlated tasks (that have different convergence rates and learning difficulties) with task-wise early stopping and task-constrained models. Further different weighting strategies of the combination of single tasks exist, i.e., Dynamic Weight Average (DWA) [40] and Dynamic Task Prioritization [27], that dynamically prioritize difficult tasks during training. The methods by [13] (GradNorm) and [20] (GCS) are based on gradients for loss scaling, while [21] adds connections between layers, and hence, these methods depend on the network architecture. [44] addresses the challenge of combining auxiliary tasks into a single coherent loss by learning (non-)linear interactions between auxiliary tasks.

3. Methodology

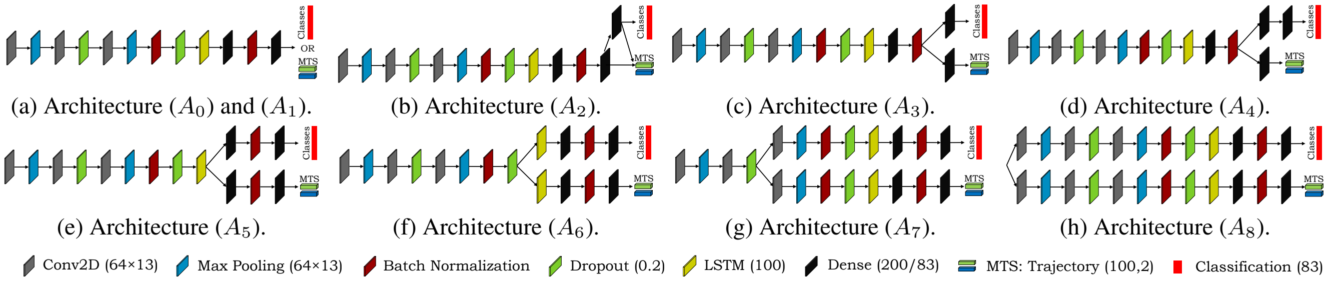
We now present the problem formulation and methodological foundation of our approach. Fig. 2 gives an overview of our method for both the inertial and visual datasets. We encode the input data sources with a CNN trunk and process the features for each individual task with separated heads. We will first describe details for the MTSC task (i.e., following the classification head) in Sec. 3.1. For the trajectory prediction task (i.e., along the regression head) we make use of *distance*, *spatio-temporal* and *distribution*-based loss functions introduced in Sec. 3.2. We then present different MTL strategies for the combination of loss functions in Sec. 3.3, and propose suitable MTL architectures in Sec. 3.4 that allow to predict the class labels and MTS trajectories. Details are given in the Appendix A.1.

3.1. Multivariate Time Series (MTS) Classification

An MTS $\mathbf{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_m\} \in \mathbb{R}^{m \times l}$ is an ordered sequence of $m \in \mathbb{N}$ streams with $\mathbf{u}_i = (u_{i,1}, \ldots, u_{i,l}), i \in \{1, \ldots, m\}$, where m is the length of the time series and l is the number of dimensions. For example in pose tracking, we might have several streams induced by sensors attached to the body plus (features extracted from) a video stream. Each MTS is associated with a class label $v \in \Omega$ from a pre-defined label set Ω. The training set is a subset of the array $\mathcal{U} = \{\mathbf{U}_1, \ldots, \mathbf{U}_n\} \in \mathbb{R}^{n \times m \times l}$, where n is the number of time series, and the corresponding labels $\mathcal{V} = \{v_1, \ldots, v_n\} \in \Omega^n$ [60]. The MTSC task is to predict an unknown class label $\hat{\mathcal{V}}$ for a given MTS. We learn the classification model using the cross-entropy loss $\mathcal{L}_{CE}(\mathcal{U}, \mathcal{V})$ [25]. For details, see Appendix A.1.

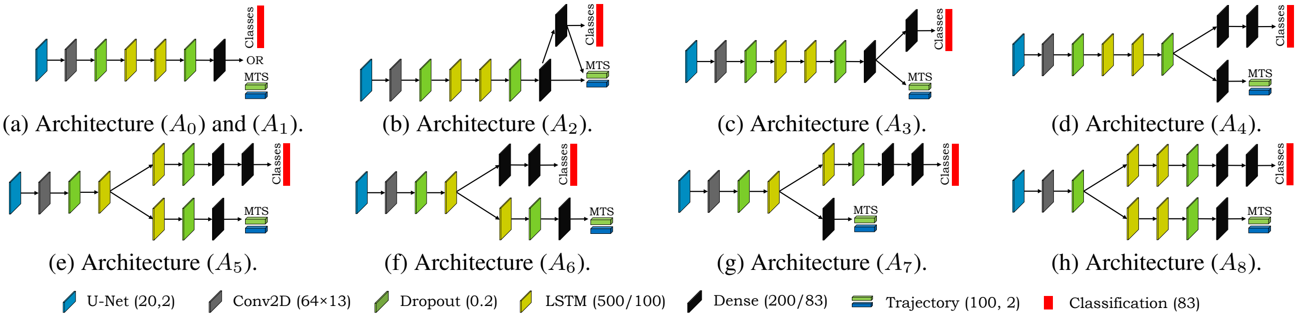
3.2. Trajectory Regression Metrics

When reconstructing or regressing trajectories, such as handwritten characters, we have to consider another multi-dimensional (discrete) time series of varying length. The MTS can take values in $\Psi \subset \mathbb{R}$ and is represented by a matrix of size $n \times d$ (n: number of timesteps, d: dimensions of the time series). Given a ground truth time series $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\} \in \mathbb{R}^{m \times d}$, the goal is to predict a time series $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{n \times d}$, such that $\mathcal{X}$ is closely aligned to $\mathcal{Y}$ [34]. For our OnHW recognition task, the prediction is of size $(100, 2)$, but can be chosen arbitrarily for different applications. In the following, we consider



(a) Architecture (A_0) and (A_1).
(b) Architecture (A_2).
(c) Architecture (A_3).
(d) Architecture (A_4).
(e) Architecture (A_5).
(f) Architecture (A_6).
(g) Architecture (A_7).
(h) Architecture (A_8).

Figure 3: Overview of **inertial**-based architectures. STL: (A_0) and (A_1). MTL (A_2) to (A_8).



(a) Architecture (A_0) and (A_1).
(b) Architecture (A_2).
(c) Architecture (A_3).
(d) Architecture (A_4).
(e) Architecture (A_5).
(f) Architecture (A_6).
(g) Architecture (A_7).
(h) Architecture (A_8).

Figure 4: Overview of **visual**-based architectures. STL: (A_0) and (A_1). MTL (A_2) to (A_8).

$\mathcal{X}$ and $\mathcal{Y}$ to be of same length n, and $\mathbf{r}_i = \mathbf{y}_i - \mathbf{x}_i$ be the residual between $\mathcal{X}_i$ and $\mathcal{Y}_i$. We consider a (differentiable) substitution-cost function $\mathcal{L} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_+$ to learn the trajectory regression task. Different loss functions have different challanges and advantages. We make use of *distance*-based, *spatio-temporal* and *distribution*-based loss functions. For more details, see Appendix A.1.

As our **distance**-based loss function we consider the mean squared error $\mathcal{L}_{MSE}(\mathcal{X},\mathcal{Y}) = \frac{1}{n}\|\mathcal{X} - \mathcal{Y}\|_2^2 = \frac{1}{n}\sum_{i=i}^{n} \mathbf{r}_i^2$ with L_2-norm $|| \cdot ||_2$, the Huber loss $\mathcal{L}_H(\mathcal{X},\mathcal{Y},\delta_H)$ [30], which is less sensitive to outliers but depends on a hyperparameter δ_H, and the Andrew's Sine loss $\mathcal{L}_{AS}(\mathcal{X},\mathcal{Y},\delta_{AS})$ [8] with hyperparameter δ_{AS}. *Distance*-based loss functions, however, do not consider relative differences in input pairs.

The following **spatio-temporal** loss functions take into account the temporal dimensions of the data and maximize the shape similarity between ground truth and predicted trajectory. The Cosine Similarity is a measure of similarity between two non-zero vectors of an inner product space. The loss is defined by $\mathcal{L}_{CS}(\mathcal{X},\mathcal{Y}) = 1 - (\mathbf{x} \cdot \mathbf{y})/(\|\mathbf{x}\|_2\|\mathbf{y}\|_2)$. Cosine Similarity is not invariant to shifts that is required for our application. The Pearson Correlation [47], in contrast, is invariant to shifts as it measures the linear relationship between two distributions in $[-1, 1]$, with 1 being a perfect alignment. Instead of the symmetric distance prediction [34], we train our model based on the Pearson Correlation loss $\mathcal{L}_{PC}(\mathcal{X},\mathcal{Y}) = \mathcal{L}_{CS}(\mathcal{X} - \overline{\mathcal{X}}, \mathcal{Y} - \overline{\mathcal{Y}})$ with $\overline{\mathcal{X}}$ and $\overline{\mathcal{Y}}$ the mean of $\mathcal{X}$ and $\mathcal{Y}$, respectively.

Finally, we also consider **distribution**-based loss functions. Specifically, we use the Wasserstein distance [23] that defines a distance between two probability distributions on a given metric space M and that represents the cost δ of an optimal mass transportation problem. To solve the learning problem, we need to minimize the loss $\mathcal{L}_{WAS_p}(\mathcal{X},\mathcal{Y})$, but calculating the gradient is computationally expensive. Hence, we optimize a smoothened Wasserstein loss function that is strictly convex [18].

3.3. Multi-Task Learning (MTL)

Our goal is to jointly classify an MTS using the cross-entropy loss and regress the corresponding trajectory, see the right part in Fig. 2. For each task, we have separate architecture heads. We show that both tasks are related, and hence, the MTL approach takes advantage of the information gain over single task approaches. The training of different tasks is non-trivial (see, [38, 39, 42, 61]). We have a set of tasks $K = \{T_1, ..., T_{|K|}\}$ with $|K|$ tasks. The naive approach is to combine losses by a weighted sum $\mathcal{L}_{total} = \sum_{i=1}^{|K|} \omega_i \mathcal{L}_i$, with pre-specified, constant weights ω_i. We use this technique as a baseline, and choose $\omega_i = 1, \forall i \in \{1, \ldots, |K|\}$ as default value, i.e., we weight the regression and classification losses equally. For trajectory regression we additionally combine two losses, namely *distance*-based metrics such as the MSE $\mathcal{L}_{MSE}$, Andrew's Sine $\mathcal{L}_{AS}$ or Huber $\mathcal{L}_H$ loss, with *spatio-temporal* distances such as the Cosine Similarity $\mathcal{L}_{CS}$ or the Pearson Correlation $\mathcal{L}_{PC}$, or *distribution*-based loss functions, i.e., the Wasserstein metric $\mathcal{L}_{WAS_p}$. We perform different weighting strategies for these losses. First, we apply the naive approach. Our second approach is to perform an epoch-dependent weighting where the weighting of the second task is dependent on the training process, i.e., the epoch number. We apply linear weight increase, squared weight increase, and squared weight decrease with respect to the weight of the second regression loss (see the blue, green and red lines in Fig. 2). As a third option, we apply DWA [40] by averaging task weighting over time. In detail, we define the weights for the current epoch e as

$$\omega_i(e) = \frac{e^{\lambda_i(e-1)/P}}{\sum_k e^{\lambda_k(e-1)/P}}; \;\; \lambda_i(e-1) = \frac{\mathcal{L}_i(e-1)}{\mathcal{L}_i(e-2)}, \quad (1)$$

where P is a pre-specified softness of task weighting. For large $P, \lambda_i \approx 1$, and tasks are weighted equally. We set $P = 1$. λ_i is the relative descending rate between previous epochs $e-1$ and $e-2$ and is in the range $(0, +\infty)$.

3.4. MTL-specific Architectures

We consider different architectures to jointly learn the two tasks. For the MTL approach, lower representation layers (trunk) have to be shared between different tasks by forking into task-specific separate layers (heads). The ratio between trunk and heads is particularly important for our application. We train the following nine architectures: only regression (A_0), only classification (A_1), and MTL-based architectures (A_2 to A_8) with different split points (from A_2 being the latest split to A_8 being the earliest split). For our experiments we implement two different feature encoders: (1) a CNN that extracts features from the channels of the IMU (see Fig. 3), and (2) a CNN that extracts features from the visual dataset (see Fig. 4). The output of the CNNs are either the class for the MTSC task, the trajectory for the regression task, or both for the MTL approach. For details, see Appendix, Sec. A.2.

4. Joint Classification and Trajectory Regression of Online Handwriting (OnHW)

There exists no state-of-the-art dataset that contains ground truth trajectories and classification labels. We recorded two novel datasets for OnHW recognition of characters with a sensor-enhanced pen written on a tablet for ground truth and three outside-in cameras for pen tip reconstruction.

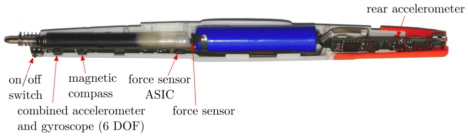
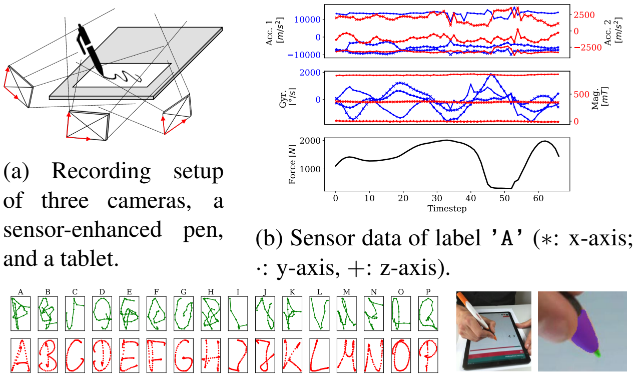


Figure 5: Pen with an integrated gyroscope, magnetometer, front and rear accelerometers, and a force sensor.



(a) Recording setup of three cameras, a sensor-enhanced pen, and a tablet.

(b) Sensor data of label `'A'` ($*$: x-axis, $\cdot$: y-axis, $+$: z-axis).

(c) Top: visual input. Bottom: ground truth trajectory interpolated to (100,2).

(d) Visual data pre-processing.

Figure 6: Data recording of our *OnHW* dataset.

Recording Setup. Our inertial dataset uses a sensor-enhanced pen [46] that contains two accelerometers (3 axes each), one gyroscope (3 axes), one magnetometer (3 axes), and one force sensor at 100 *Hz* (Fig. 5). We replace the pencil lead with a Wacom EMR module and record ground truth trajectories at 30 *Hz* on a Samsung Galaxy Tab S4 tablet. Our visual dataset uses three cameras pointing on the pen (Fig. 6a) to record the movement of the pen tip at 60 *Hz*. A right-handed person wrote ≈18×83 characters containing small (26) and capital (26) letters, numbers (10), and symbols (21).

Pre-processing and Dataset. Fig. 6b shows an IMU signal of the letter `'A'` (from the inertial dataset) and Fig. 6c shows characters `'A'` to `'P'` (from the visual dataset) from the pre-computed trajectory in camera coordinates (top row) and the ground truth trajectory produced by the tablet (bottom row). For the camera-based trajectories we segment the pixels of 100 random images of the dataset in the classes "pen" (the purple parts of the pen in Fig. 6d), "pen tip" (green parts of the pen), and "background". We train U-Net [49] to predict the pen tip from all images and choose the middle 90*th* percentile of 20 top-left pen tip pixels as the trajectory in camera coordinates (Fig. 6c). We interpolate the ground truth trajectory to (100,2). A 71/29 train/validation split results in 822 training and 332 validation characters for the IMU dataset, and in 2,466 training and 992 validation characters for the visual dataset. Datasets and source code publicly available upon publication.[1]

CNN Architectures. The visual time series input is of size $m = 40$ due to 20 pixels in both camera axes, the length l is variable and dependent on the length of the character. For the IMU input the time series is of size $m = 13$ (for the two accelerometers, the gyroscope and the magnetometer

[1] Dataset and source code: https://iis.fraunhofer.de/onhw-dataset/

Network	MSE+CE		AS+CE		H+CE		MSE+PC+CE		MSE+CS+CE		MSE+WAS+CE		PC+CE	CS+CE	WAS+CE
	Traj.	**Class.**	**Traj.**	**Class.**	**Traj.**	**Class.**	**Traj.**	**Class.**	**Traj.**	**Class.**	**Traj.**	**Class.**	**Class.**	**Class.**	**Class.**
Only regression (A_0)	0.1705	-	**0.1594**	-	0.1501	-	0.1723	-	1.0023	-	0.3107	-	-	-	-
Only classification (A_1)	-	88.11	-	-	-	-	-	-	-	-	-	-	-	-	-
Class. for regr. (A_2)	0.1169	86.69	0.1779	9.78	**0.1290**	62.78	**0.1127**	86.81	0.3554	86.28	0.1612	7.28	86.73	86.02	12.60
Latest split (A_3)	0.1381	86.67	0.1856	49.31	0.1569	66.73	**0.1381**	85.75	6.1464	87.22	0.3375	20.79	86.22	84.15	25.53
Late split (A_4)	0.1372	86.46	0.1421	76.28	0.1581	63.64	**0.1357**	**88.64**	1.3928	87.62	0.3262	26.65	86.67	**89.51**	29.74
Split after LSTM (A_5)	0.1370	87.34	0.1629	68.64	0.1458	73.74	**0.1386**	85.53	1.0578	**88.58**	0.3284	35.49	86.93	88.03	54.84
Split after 2. Drop. (A_6)	0.1623	87.68	0.1464	83.96	0.1580	84.76	**0.1647**	84.94	1.0053	85.28	0.3208	80.59	83.37	84.49	84.65
Split after 1. Drop. (A_7)	0.1866	84.27	0.1676	**86.93**	0.1546	86.89	**0.1638**	84.55	1.1388	87.20	0.3071	84.13	83.58	86.34	81.87
Separate heads (A_8)	0.1936	86.87	0.1660	86.02	**0.1533**	**88.43**	**0.1490**	87.03	1.0986	85.79	0.3315	82.03	87.15	**88.15**	82.58

Table 2: Evaluation results for the IMU-based dataset trained with different loss combinations. Trajectory evaluation metric: root mean squared error (RMSE). Classification accuracy given in %. A_0 and A_1: single task architectures. A_2 to A_8: MTL architectures. Underlined: baselines. **Bold**: best results. Columns are combinations of different loss functions, e.g., MSE+CE is a combination of the $\mathcal{L}_{MSE}$ and the $\mathcal{L}_{CE}$ losses, etc.

Network	MSE+CE		AS+CE		H+CE		MSE+PC+CE		MSE+CS+CE		MSE+WAS+CE		PC+CE	CS+CE	WAS+CE
	Traj.	**Class.**	**Traj.**	**Class.**	**Traj.**	**Class.**	**Traj.**	**Class.**	**Traj.**	**Class.**	**Traj.**	**Class.**	**Class.**	**Class.**	**Class.**
Only regression (A_0)	0.1360	-	0.1388	-	0.1271	-	0.1348	-	2.5733	-	0.2302	-	-	-	-
Only classification (A_1)	-	73.19	-	-	-	-	-	-	-	-	-	-	-	-	-
Class. for regr. (A_2)	**0.1250**	**80.15**	0.1279	12.47	0.1475	55.81	0.1327	77.27	0.4900	74.33	0.1844	56.44	74.85	73.52	71.10
Latest split (A_3)	0.4314	23.21	0.1327	74.3	0.1577	64.15	**0.1401**	**75.54**	0.6713	9.32	0.1960	49.22	76.57	10.34	75.08
Late split (A_4)	0.1351	78.49	0.1260	78.23	**0.1230**	79.42	**0.1284**	**80.89**	1.1177	77.08	0.1348	74.50	58.07	74.42	74.25
Split after LSTM (A_5)	0.1291	**80.40**	**0.1252**	**81.24**	0.1307	78.06	0.1478	73.21	0.1705	77.22	0.1359	76.64	75.17	79.21	76.47
LSTM in tr. head (A_6)	0.1334	77.33	0.1383	74.31	0.1605	63.31	**0.1243**	**81.64**	0.1747	78.80	0.1376	73.64	74.05	77.47	75.42
LSTM in cl. head (A_7)	0.1378	78.58	0.1267	80.52	0.1330	78.86	0.1459	74.25	1.2483	79.99	0.1380	75.87	**80.51**	72.10	78.03
Split after 1. Drop. (A_8)	**0.1246**	78.56	0.1248	75.69	0.1310	75.18	0.1251	**79.27**	0.4132	78.67	0.2448	75.83	78.83	**80.76**	**79.42**

Table 3: Evaluation results for the visual dataset trained with different loss combinations. For definitions, see Table 2.

with 3 axes each plus the force sensor) with a variable length l. When training both encoders, each batch is bias shuffled, such that each batch contains letters of approximately the same time step length. To account for variable batch length, we use zero padding to the maximal size in each batch. For the classification task the last *dense* layer has 83 neurons, for the trajectory regression task 200 neurons (reshaped: 100×2). In a preliminary study, we searched for the optimal dropout and LSTM neurons for the visual CNN architecture (see Appendix, Fig. 11). We choose an LSTM combination of 500 and 100 units, and two dropout layers with 20% dropout after the convolution and the second LSTM layer.

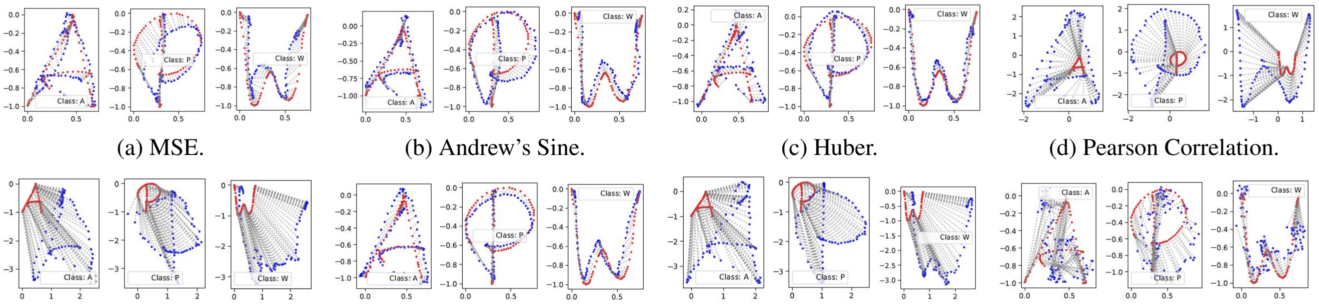
5. Experimental Results

We here describe the results for the IMU dataset (Sec. 5.1 to 5.3, Table 2) and for the visual dataset (Sec. 5.4, Table 3). More specifically, Sec. 5.1 compares the proposed MTL architectures against the baseline networks A_0 and A_1. We use the cross-entropy loss for MTSC, and *distance*, *spatio-temporal*, and *distribution*-based losses for trajectory prediction (Sec. 5.2). In Sec. 5.3, we evaluate our MTL strategies, and compare our methods to state-of-the-art techniques in Sec. 5.5.

Preliminary. We first want to emphasize that improvements in the smoothness of the predicted trajectory are utmost importance in our handwriting application, but that smoother trajectories do not necessarily lead to a significant improvement of the reconstruction error. This is similar to image reconstruction, where the biggest gain in performance is achieved by vaguely reconstructing the image, yet the image still looks unnatural for humans. Hence, also small trajectory improvements are of interest.

Hardware and Training Setup. For all experiments we use Nvidia Tesla V100-SXM2 GPUs with 32 GB VRAM equipped with Core Xeon CPUs and 192 GB RAM. We use the vanilla Adam optimizer with learning rate 10^{-4}. We run each experiment three times for 20,000 epochs with a batch size of 50 and report averaged results (over five epochs).

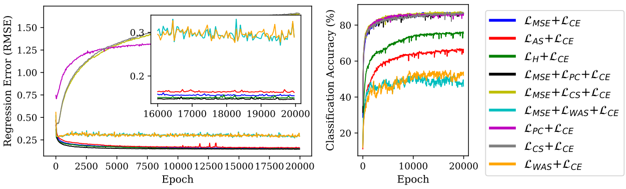
Evaluation Metric. For evaluation we compute the categorical metric for class prediction in %, and the root mean squared error (RMSE) for trajectory prediction. As in [59], we also compute the *geometric shape*-based Fréchet [10], Hausdorff [55] and Procrustes [51] measures, as well as the *time warping* approach DTW [15] (see Sec. 2.2). Due to a correlation between these metrics with the RMSE and for better readability, we only report the RMSE.



(a) MSE. (b) Andrew's Sine. (c) Huber. (d) Pearson Correlation.

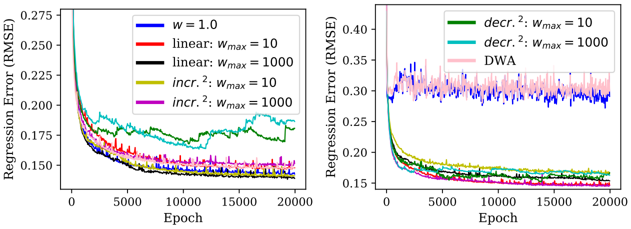
(e) Cosine Similarity. (f) MSE + Pearson Correlation. (g) MSE and Cosine Similarity. (h) MSE + Wasserstein.

Figure 7: Trajectory prediction (blue) against the ground truth trajectory (red) of the characters `'A'`, `'P'` and `'W'` based on inertial data trained with different combinations of loss functions.



(a) Validation error (RMSE). (b) Validation accuracy (%).

Figure 8: Evaluation of combinations of loss functions averaged over all architectures (A_0-A_8). MTL strategy: naive weighting $w = 1$. Baseline (blue): $\mathcal{L}_2 + \mathcal{L}_{CE}$.



(a) $\mathcal{L}_{MSE}$, $\mathcal{L}_{PC}$ and $\mathcal{L}_{CE}$. (b) $\mathcal{L}_{MSE}$, $\mathcal{L}_{WAS_1}$ and $\mathcal{L}_{CE}$.

Figure 9: MTL strategy evaluation for $\mathcal{L}_{MSE}$ combined with *spatio-temporal* $\mathcal{L}_{PC}$ and *distribution*-based $\mathcal{L}_{WAS_1}$ losses averaged over all architectures (baseline: blue).

5.1. MTL Architecture Evaluation

As a baseline for the inertial dataset, model A_0 results in an error of 0.1705 using the $\mathcal{L}_{MSE}$, and model A_1 in an accuracy of 88.11%. Exemplary reconstructed letters of the baseline method is shown in Fig. 7a. The trajectory regression task improves up to 0.1169 for model A_2 to 0.1623 for model A_6, but decreases for model A_7 and A_8. The accuracy of 86.69% (A_2) is less accurate than the baseline A_1 (87.68%). We conclude, that a late split has a positive influence on the trajectory regression by sharing more trainable parameters in the trunk. This even holds for smaller model sizes (see Appendix, Table 5). For a larger regression head (A_7 and A_8) the model is prone to overfitting.

5.2. Loss Function Evaluation

Single Loss Functions. For more details on hyperparameter searches, see Appendix Sec. A.3. For the *distance*-based metrics, i.e., Andrew's Sine and Huber we observe a decrease in trajectory error (Fig. 7b and 7c) compared to $\mathcal{L}_{MSE}$ at the cost of a deterioration of the classification accuracy. For the $\mathcal{L}_{AS}$ loss, the trajectory error decreases for models A_6 to A_8 against $\mathcal{L}_{MSE}$, while the classification accuracy decreases. The same holds for the $\mathcal{L}_H$ loss, where we can increase the classification accuracy up to 88.43%. Using *spatio-temporal* losses we are able to learn the shape of the characters (Fig. 7d and 7e), but at a wrong scale. A trajectory trained with the $\mathcal{L}_{PC}$ is smoother (less variance) compared to $\mathcal{L}_{CS}$, yet with a lower accuracy in the classification. Our goal is to minimize the distance of the predicted trajectory while ensuring a smooth shape. Hence, we train the (distance) $\mathcal{L}_{MSE}$ loss combined with the (spatio-temporal) $\mathcal{L}_{PC}$ and $\mathcal{L}_{CS}$.

Combined Loss Functions. Fig. 8 compares all metrics based on the naive weighting. For the combination of $\mathcal{L}_{MSE}$ and $\mathcal{L}_{PC}$, the regression loss improves over a model trained on $\mathcal{L}_{MSE}$ only, while providing a smoother trajectory (Fig. 7f). The combined loss $\mathcal{L}_{MSE} + \mathcal{L}_{CS}$ does not scale the characters correctly (Fig. 7g) and gives a less smooth trajectory than the approach with either only $\mathcal{L}_{MSE}$ (Fig. 7a), only $\mathcal{L}_{PC}$ (Fig. 7d), or only $\mathcal{L}_{CS}$ (Fig. 7e). We evaluate the $\mathcal{L}_{WAS_p}$ loss when combined with the $\mathcal{L}_{MSE}$ loss. For larger p, the predictions become more evenly distributed (as found by [23]). When choosing $p = 1$ and the naive weighting strategy (Fig. 9b) the trajectory prediction error is large (Fig. 7h), but can be significantly improved using alternative MTL strategies, as described in the following.

5.3. MTL Strategy Evaluation

We now compare all described MTL strategies (Sec. 3.3): naive weighting (w=1), linear as well as squared increase, squared weight decrease with maximal weighting of w_{max}=$\{10, 1000\}$, and DWA [40]. Fig. 9 shows MTL results for $\mathcal{L}_{MSE}$ combined with $\mathcal{L}_{PC}$ (Fig. 9a) and $\mathcal{L}_{WAS_p}$ with $p = 1$ (Fig. 9b). When combining $\mathcal{L}_{WAS_1}$ with the $\mathcal{L}_{PC}$ loss, the error further decreases for linear and squared weight increase. Squared weight decrease notably increases the error as the scaling of the shape diverges. With an optimal epoch-dependent weighting strategy the Pearson Correlation provides a suitable shape while still yielding to an improvement in accuracy. Combined with the $\mathcal{L}_{WAS_1}$ loss, all weighting strategies significantly decrease the error (improvements between 0.16 to 0.18) in comparison to the naive approach (0.31), yet still preserve smooth predictions. When only training based on the $\mathcal{L}_{MSE}$ loss, the distance is still smaller, but predicted trajectories are less smooth and not realistic. For the $\mathcal{L}_{PC}$ loss combined with the $\mathcal{L}_{MSE}$ loss DWA is worse than the linear or squared weight increase. As the loss of $\mathcal{L}_{WAS_p}$ is higher than the loss of $\mathcal{L}_{MSE}$, DWA cannot optimize the training (Fig. 9b). We conclude, that the order of weight increase is important for combining metrics and that a slow weight increase is the best approach for jointly learning the classification and trajectory regression tasks.

5.4. Visual Dataset Evaluation

Loss Functions. The visual dataset is more challenging than the IMU dataset as the network needs to learn the transformation from the camera to the tablet coordinates and needs to identify the pen tip hover and touch data between strokes. This results in a much larger computing times, i.e., an average run time of 5.4s per epoch (see Appendix, Table 7). The baselines yield an error of 0.1360 for the trajectory regression task and 73.19% accuracy for the MTSC task (see Table 3). For the evaluation of MTL architectures, we can draw the same conclusions as in Sec. 5.1. Similar to the IMU dataset, for both, the $\mathcal{L}_H$ and the $\mathcal{L}_{AS}$ loss we (partly) observe a decrease in trajectory error at the cost of a worsening classification accuracy. The single *spatio-temporal* $\mathcal{L}_{PC}$ loss increases the classification accuracy (80.51%), but yields improperly scaled characters. Through the combination of $\mathcal{L}_{MSE} + \mathcal{L}_{PC}$, we further decrease the trajectory error (0.1243) and increase the classification accuracy (81.64%) for architecture A_6, while still providing a smooth trajectory. Neither the single *distribution*-based $\mathcal{L}_{WAS_p}$ loss nor the combination with the $\mathcal{L}_{MSE}$ loss can be used to improve the single and combined tasks. For more details, see Appendix, Sec. A.3.

Method	Inertial	Visual
MLSTM-FCN [31] w/ SE	89.33%	80.49%
w/o SE	88.41%	78.73%
TapNet [60]	89.02%	79.27%
Ours	**89.51%**	**81.64%**
	(A_4, CS + CE)	(A_6, MSE + PC + CE)

Table 4: Comparison of state-of-the-art MTSC methods with our approach on the inertial and visual datasets.

5.5. Comparison to MTSC Methods

We compare our method with MLSTM-FCN [31] and TapNet [60], two approaches achieving the highest classification accuracy on the well-known UEA MTS dataset [5]. For both techniques we interpolate the IMU data to 114 and the visual data to 71 timesteps. We evaluate various configurations of both networks in the line with suggestions by the authors. We optimized TapNet and searched for different configurations (with default parameters for the CNN, LSTM and random projection) and optimized the hyperparameters for training (such as different learning rates at 10^{-3}, 10^{-4} and 10^{-5}). For the MLSTM-FCN we achieved the best results with its squeeze-and-excitation block (see Table 4). TapNet achieves a better MTSC accuracy in comparison to MLSTM-FCN for the visual dataset. However, this still lacks behind our best reported results, which are both higher than TapNet and MLSTM-FCN. For the visual dataset our approach shows a larger margin and is notably better for certain loss combinations and architectures.

6. Conclusion

We addressed the problem of aligning a ground truth trajectory that is smooth over time by combining *distance*-based with *spatio-temporal* and *distribution*-based metrics. For the application of OnHW recognition of characters based on IMU and visual data sources, we significantly improved the trajectory prediction task, while still providing similar shapes. We proposed a framework of architectures and evaluated different MTL techniques. Through the combination of the trajectory prediction task with the subtly correlated MTS classification task we further improved the classification accuracy.

Acknowledgements

This work was supported by the Federal Ministry of Education and Research (BMBF) of Germany by Grant No. 01IS18036A (David Rügamer) and by the research program Human-Computer-Interaction through the project "Schreibtrainer", Grant No. 16SV8228, as well as by the Bavarian Ministry for Economic Affairs, Infrastructure, Transport and Technology through the Center for Analytics-Data-Applications (ADA-Center) within the framework of "BAYERN DIGI-TAL II".

References

[1] Md. Shahinur Alam, Ki-Chul Kwon, Md. Ashraful Alam, Mohammed Y. Abbass, Shariar Md Imtiaz, and Nam Kim. Trajectory-Based Air-Writing Recognition Using Deep Neural Network and Depth Sensor. In *Sensors*, volume 20(376), Jan. 2020.

[2] Fevzi Alimoglu and Ethem Alpaydin. Combining Multiple Representations and Classifiers for Pen-based Handwritten Digit Recognition. In *Intl. Conf. on Document Analysis and Recognitino (ICDAR)*, volume 2, Ulm, Germany, Aug. 1997.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *Intl. Conf. on Machine Learning (ICML)*, volume 70, pages 214–223, 2017.

[4] J. Arunnehru, A. K. Nandhana Davi, R. Raghul Sharan, and Poornima G. Nambiar. Human Pose Estimation and Activity Classification Using Machine Learning Approach. In *Intl. Conf. on Soft Computing and Signal Processing (ICSCSP)*, pages 113–123, Mar. 2020.

[5] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, and Paul Southam Eamonn Keogh. The UEA Multivariate Time Series Classification Archive, 2018. In *arXiv preprint arXiv:1811.00075*, Oct. 2018.

[6] Jonathan T. Barron. A General and Adaptive Robust Loss Function. In *Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4326–4334, Long Beach, CA, June 2019.

[7] Vasileios Belagiannis, Christian Rupprecht, Gustavo Carneiro, and Nassir Navab. Robust Optimization for Deep Regression. In *Intl. Conf. on Computer Vision (ICCV)*, pages 2830–2838, Santiago de Chile, Chile, Dec. 2015.

[8] Michael J. Black and Anand Rangarajan. On the Unification of Line Processes, Outlier Rejection, and Robust Statistics with Applications in Early Vision. In *Intl. Journal of Computer Vision (IJCV)*, volume 19, page 57–91, July 1996.

[9] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-Aware Learning of Maps for Camera Localization. In *Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2616–2625, 2018.

[10] Karl Bringmann. Why Walking the Dog Takes Time: Fréchet Distance has no Strongly Subquadratic Algorithms unless SETH fails. In *Symp. on Foundations of Computer Science (FOCS)*, Apr. 2014.

[11] Xingyu Cai, Tingyang Xu, Jin-Feng Yi, Junzhou Huang, and Sanguthevar Rajasekaran. DTWNet: a Dynamic Time Warping Network. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[12] Junshen Kevin Chen, Wanze Xie, and Yutong (Kelly) He. Motion-based Handwriting Recognition. In *arXiv preprint arXiv:2101.06022*, Jan. 2021.

[13] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. In *Intl. Conf. on Machine Learning (ICML)*, volume 80, pages 794–803, 2018.

[14] Samuel Cohen, Giulia Luise, Alexander Terenin, Brandon Amos, and Marc Peter Deisenroth. Aligning Time Series on Incomparable Spaces. In *Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, volume 130, pages 1036–1044, 2021.

[15] Marco Cuturi and Mathieu Blondel. Soft-DTW: a Differentiable Loss Function for Time-Series. In *Intl. Conf. on Machine Learning (ICML)*, volume 70, pages 894–903, 2017.

[16] ShaiBen David and Reba Schuller. Exploiting Task Relatedness for Multiple Task Learning. In *Learning Theory and Kernel Methods*, volume 2777, pages 567–580, 2003.

[17] Thomas Deselaers, Daniel Keysers, and Henry A. Rowley. GyroPen: Gyroscopes for Pen-Input with Mobile Phones. In *Trans. on Human-Machine Systems*, Apr. 2015.

[18] Prafulla Dhariwal and Jeevana Inala. The Wasserstein Loss Function. Dec. 2015.

[19] Ian L. Dryden and Kanti V. Mardia. Statistical Shape Analysis with Applications in R. In *Wiley Series in Probability and Statistics*, 2016.

[20] Yunshu Du, Wojciech M. Czarnecki, Siddhant M. Jayakumar, Mehrdad Farajtabar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting Auxiliary Losses Using Gradient Similarity. In *arXiv preprint arXiv:1812.02224*, Nov. 2020.

[21] Yuchun Fang, Zhengyan Ma, Zhaoxiang Zhang, Xu-Yao Zhang, and Xiang Bai. Dynamic Multi-Task Learning with Convolutional Neural Network. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1668–1674, 2017.

[22] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep Learning for Time Series Classification: a Review. In *Data Mining and Knowledge Discovery*, July 2019.

[23] Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya-Polo, and Tomaso Poggio. Learning

with a Wasserstein Loss. In *Advances in Neural Information Processing Systems (NIPS)*, volume 2, page 2053–2061, Dec. 2015.

[24] Damien Garreau, Rémi Lajugie, Sylvain Arlot, and Francis Bach. Metric Learning for Temporal Sequence Alignment. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1817–1825, 2014.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. In *MIT Press*, 2016.

[26] Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor Forcing: A New Algorithm for Training Recurrent Networks. In *Advances in Neural Information Processing Systems (NIPS)*, Barcelona, Spain, 2016.

[27] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic Task Prioritization for Multitask Learning. In *Europ. Conf. on Computer Vision (ECCV)*, pages 282–299, 2018.

[28] David Ha and Douglas Eck. A Neural Representation of Sketch Drawings. In *Intl. Conf. on Learning Representations (ICLR)*, Vancouver, BC, Canada, Apr. 2018.

[29] Yifan Hao and Huiping Cao. A New Attention Mechanism to Classify Multivariate Time Series. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1999–2005, 2020.

[30] Peter J. Huber. Robust Statistics. In *John Wiley and Sons*, New York, NY, 1981.

[31] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate LSTM-FCNs for Time Series Classification. In *Neural Network*, volume 116, pages 237–245, Aug. 2019.

[32] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7482–7491, Salt Lake City, UT, 2018.

[33] Christopher Koellner, Marc Kurz, and Erik Sonnleitner. What Did You Mean? An Evaluation of Online Character Recognition Approaches. In *Intl. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–6, 2019.

[34] Georg Kohl, Kiwon Um, and Nils Thuerey. Learning Similarity Metrics for Numerical Simulations. In *Intl. Conf. on Machine Learning (ICML)*, Vienna, Austria, Feb. 2020.

[35] Orhan Konak, Pit Wegner, and Bert Arnrich. IMU-Based Movement Trajectory Heatmaps for Human Activity Recognition. In *Sensors*, volume 20, Dec. 2020.

[36] Tuomo Korenius, Jorma Laurikkala, and Martti Juhola. On Principal Component Analysis, Cosine and Euclidean Measures in Information Retrieval. In *Information Science*, volume 177(22), pages 4893–4905, Nov. 2007.

[37] Vijay Kumar, Jitender Kumar Chhabra, and Dinesh Kumar. Performance Evaluation of Distance Metrics in the Clustering Algorithms. In *INFOCOMP*, volume 13(1), pages 38–51, June 2014.

[38] Changsheng Li, Junchi Yan, Fan Wei, Weishan Dong, Qingshan Liu, and Hongyuan Zha. Self-Paced Multi-Task Learning. In *Intl. Conf. on Association for the Advancement of Artificial Intelligence (AAAI)*, pages 2175–2181, 2017.

[39] Sicong Liang and Yu Zhang. A Simple General Approach to Balance Task Difficulty in Multi-Task Learning. In *arXiv preprint arXiv:2002.04792*, Feb. 2020.

[40] Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-End Multi-Task Learning with Attention. In *Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1871–1880, 2019.

[41] Shuangjun Liu, Naveen Sehgal, and Sarah Ostadabbas. Adapted Human Pose: Monocular 3D Human Pose Estimation with Zero Real 3D Pose Data. In *arXiv preprint arXiv:2105.10837*, May 2021.

[42] Keerthiram Murugesan and Jaime Carbonell. Self-Paced Multitask Learning with Shared Knowledge. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2522–2528, 2017.

[43] Nika Naghtalab, Cameron Musco, and Bo Waggoner. Toward a Characterization of Loss Functions for Distribution Learning. In *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, BC, Canada, 2019.

[44] Aviv Navon, Idan Achituve, Haggai Maron, Gal Chechik, and Ethan Fetays. Auxiliary Learning by Implicit Differentiation. In *Intl. Conf. on Learning Representations (ICLR)*, 2021.

[45] Dvir Ben Or, Michael Kolomenkin, and Gil Shabat. Generalized Quantile Loss for Deep Neural Networks. In *arXiv preprint arXiv:2012.14348*, Dec. 2020.

[46] Felix Ott, Mohamad Wehbi, Tim Hamann, Jens Barth, Björn Eskofier, and Christopher Mutschler. The OnHW Dataset: Online Handwriting Recognition from IMU-Enhanced Ballpoint Pens with Machine Learning. In *Proc. of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, volume 4(3), article 92, Cancún, Mexico, Sept. 2020.

[47] Karl Pearson. Notes on the History of Correlation. In *Biometrika*, volume 13(1), pages 25–45, Oct. 1920.

[48] Aaditya Ramdas, Nicolás García Trillos, and Marco Cuturi. On Wasserstein Two-Sample Testing and Related Families of Nonparametric Tests. In *Entropy*, volume 19(47), Jan. 2017.

[49] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, LNCS*, volume 9351, pages 234–241, 2015.

[50] Maximilian Schrapel, Max-Ludwig Stadler, and Michael Rohs. Pentelligence: Combining Pen Tip Motion and Writing Sounds for Hadnwritten Digit Recognition. In *Conf. on Human Factors in Computing Systems*, number 131, pages 1–11, Apr. 2018.

[51] Peter H. Schönemann. A Generalized Solution of the Orthogonal Procrustes Problem. In *Psychometrika*, volume 31(1), pages 1–10, Mar. 1966.

[52] Ali Seyed Shirkhorshidi, Saeed Aghabozorgi, and Teh Ying Wah. A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data. In *PLOS ONE*, volume 10(12), Dec. 2015.

[53] Anuj Srivastava and Eric P. Klassen. Functional and Shape Data Analysis. In *Springer Series in Statistics*, 2016.

[54] Marc Strickert, Frank-Michael Schleif, and Udo Seiffert. Derivatives of Pearson Correlation for Gradient-based Analysis of Biomedical Data. In *Inteligencia Artificial*, volume 12(37), pages 37–44, Mar. 2008.

[55] Abdel Aziz Taha and Allan Hanbury. An Efficient Algorithm for Calculating the Exact Hausdorff Distance. In *Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, volume 37(11), Nov. 2015.

[56] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. In *Journal of Machine Learning Research (JMLR)*, pages 2579–2605, Nov. 2008.

[57] Cédric Villani. Optimal Transport, Old and New. In *Springer*, Dec. 2006.

[58] Jeen-Shing Wang, Yu-Liang Hsu, and Cheng-Ling Chu. Online Handwriting Recognition Using and Accelermeter-Based Pen Device. In *Intl. Conf. on Computer Science and Engineering (CSE)*, 2013.

[59] Hanyuan Zhang, Xingyu Zhang, Qize Jiang, Baihua Zheng, Zhenbang Sun, Weiwei Sun, and Changhu Wang. Trajectory Similarity Learning with Auxiliary Supervision and Optimal Matching. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 3209–3215, 2020.

[60] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. TapNet: Multivariate Time Series Classification with Attentional Prototypical Network. In *Intl. Conf. on Association for the Advancement of Artificial Intelligence (AAAI)*, pages 6845–6852, 2020.

[61] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial Landmark Detection by Deep Multi-Task Learning. In *Europ. Conf. on Computer Vision (ECCV)*, pages 94–108, 2014.

[62] PYi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J. Leon Zhao. Exploiting Multi-Channels Deep Convolutional Neural Networks for Multivariate Time Series Classification. In *Frontiers of Computer Science*, volume 10, pages 96–112, Sept. 2015.